

Courant Institute of
Mathematical Sciences

AEC Computing and Applied Mathematics Center

Engineering Aspects of High
Speed Data Distribution

Frederic Ruben

AEC Research and Development Report

TID-4500

MATHEMATICS AND COMPUTERS

March 1, 1966



New York University

NYO-1480-48
C.1

UNCLASSIFIED

AEC Computing and Applied Mathematics Center
Courant Institute of Mathematical Sciences
New York University

TID-4500

NYO-1480-48
Mathematics and Computers

ENGINEERING ASPECTS OF HIGH SPEED
DATA DISTRIBUTION

by

Fredric Ruben

March 1, 1966

Contract No. AT(30-1)-1480

UNCLASSIFIED

TABLE OF CONTENTS

	Page
1. Introduction	3
2. Computer Hardware Specifications	4
3. Memory Configuration	6
4. Memory Selection	11
5. Memory Request Conflicts	13
6. Conflict Checker	14
7. Memory Addressing - The Time-Sharing System	17
8. The Conflict Checker	22
9. Trunk-Bank Selection	23
10. Data Distribution	25
11. Data and Address Distribution Belt Operation	33
12. Data Distribution Belt Limitations	34
13. Conclusions	36

LIST OF TABLES

Table 1.	Size-Speed Relations for 16×10^6 Words Operating at 10^9 Words/Sec.	8
----------	---	---

LIST OF ILLUSTRATIONS

Figure 1.	Limitations of Present-Day Core Memories	7
Figure 2.	Random Access Memory Cost, 1964	9
Figure 3.	Central Memory (64 Reservoirs, 64 Banks/Reservoir, 4096 Words/Bank, 64 Bits/Word)	10
Figure 4.	Total Number of Gates vs Number of Simultaneous Requests for Conflict Checker	15
Figure 5A.	Memory Address Selection Block Diagram	18
Figure 5B.	Memory Addressing 1 "Time-Sharing" Systems (1 -Bit)	19
Figure 6.	Basic Address Time-Sharing Logic Block	20
Figure 7.	Long and Short Path Switching Network	21
Figure 8.	Driver Fan-Out Module	23
Figure 9.	Trunk Conflict Checker	24
Figure 10.	Trunk-Bank Selection (Sheet 1)	26
	Trunk-Bank Selection (Sheet 2)	27
Figure 11.	Data Distribution Belt Block Diagram (1 -Bit)	28
Figure 12.	Address Belt Coincidence Generation (1 -Bit)	29
Figure 13.	Data and Address Belt Alert Generation	31
Figure 14.	Data Distribution Belt (1 -Bit)	32

ENGINEERING ASPECTS OF HIGH SPEED DATA DISTRIBUTION

1. INTRODUCTION

Continuing experience in computer usage has indicated a need for large capacity computing machines capable of operating at very high speeds. That this need is compelling is testified to by an increased interest stimulated by the multitude of calculations presently infeasible of being performed on available computing machinery. The limitations of present-day equipment in memory size and distribution rates have precluded the undertaking of such calculations. Problem types for which such machines would be applicable are:

- 1) Integration of partial differential equations
- 2) Hydrodynamic and energy transfer calculations
- 3) Long range numerical weather forecasting
- 4) Matrix calculations
- 5) Radar signal processing including:
 - a) tracking
 - b) correlation
 - c) track initiation
 - d) prediction
- 6) Matched filters and pulse compression
- 7) Radar control
- 8) Seismometer signal processing
- 9) Machine code optimization
- 10) Adaptive system simulation
- 11) List processing
- 12) Picture-processing problems such as bubble chamber photograph analysis, picture decoding in general, etc.

Such calculations may be undertaken by computers which can be classified as "Parallel Computing Machines". Several existing computers tend toward parallel processing but in fact fall short of being truly parallel computers because of their limitations dictated by geometric structure, inability to check for logical independence of operations involving large numbers of memory locations, or inefficient access to the common central memory. (i.e. Solomon, CDC 6600, IBM 360/92).^{1,2}

It is anticipated that computational speeds of the order suggested for parallel processing (10^9 instructions/sec)³ will be achieved only through the efficient distribution of data from a central memory or between processing elements. This report proposes a system for efficient and rapid data selection and distribution.

2. COMPUTER HARDWARE SPECIFICATIONS

Preliminary investigation into the software attendant to parallel computation has revealed a need for very large memory capacity and very high data transfer rates. It has been suggested that a computer to be used in such a manner would contain 16×10^6 words of central memory storage, 64 processing units, and data transfer rates from central memory in the order of 10^9 words per second with word lengths of 64 bits.⁴ In addition, "core storage should be uniformly addressable, any processing unit (CPU) should be able to address any word within central memory (CM) either for data or for instructions, and hardware should be sufficiently modular so that defective subassemblies could be replaced with minimal interruption of operation".³

A close examination of the specifications set forth reveals that a computer of such size may not be feasible. The requirement for 16×10^6 words of core storage implies 10^9 bits of memory. This, in turn, implies a minimum of 10^9 cores. It has been estimated that to produce a memory containing 10^9 cores by automatic machines making and wiring elements at the rate of one per second would require more than 30 years to complete.⁵ Even if this estimate were incorrect, the economics of such an undertaking may be prohibitive.

At present day prices, 4096 words of core memory cost in the neighborhood of \$15K to \$20K. This would fix the cost per bit at about \$.05 to \$.06. These costs can be significantly reduced if large quantities are purchased. Assuming the cost per bit can be reduced to \$.01, then the cost of the suggested memory would be approximately $\$10 \times 10^6$. If this expenditure represents say one-third to one-half the total cost of the

memory system, then we are talking of an expenditure of $\$20 \times 10^6$ or $\$30 \times 10^6$. This does not include costs for such things as CPU's, display systems, controllers, etc.

Far more cogent than delivery time or cost is the engineering difficulties encountered by the physical size of such a memory. In order to achieve data transfer rates of the magnitude suggested, multiple words need be accessed simultaneously. This implies that memory locations be equidistant from a central interrogating device. Certain latitudes are of course permitted, but an attempt should be made to keep differences in transmission distances to a minimum. If we envision transmission of 16 words every 15 nanoseconds (1.07×10^9 words/sec), then transmission time differences should not exceed 1.5 nanoseconds. Even this deviation may be too large since it represents 10% of the time between transmissions. At any rate, if point-to-point transmissions are achieved by co-axial cable, then this time represents a difference of one foot between distances from nearest and farthest transmitters to receiver.

Previous work done by the Engineering Research Laboratory at N.Y.U. included a study of very high density magnetic core plane configurations.⁶ It was found that using 30 mil O.D. cores, 5 mils thick, a packing density of 128 cores per inch could be obtained. On this basis, to achieve 4096 words of 64 bits each, a volume of 1.344x1.92x.512 inches would be required. As a result, if each of these modules represents one bank of memory, then 64 such modules placed end-to-end on their shortest dimension would require 32 inches. Even this distance seems to be in excess of what is needed.

It is not to be expected that physical dimensions required by the proposed memory system will be approached with magnetic cores even with improved packaging techniques. It is therefore apparent that either non-magnetic core memories are used or a reduction in data transmission rates is allowed.

The use of non-magnetic core memories is not intractable. Recent advances made by manufacturers indicate that such memories will compete favorably with the added advantage of small size. In fact, it has been suggested that with improvements in technology, future computers may be batch-fabricated with reductions in size necessitating "throw-away" units.⁷ Work done by the Engineering Research Laboratory in cylindrical thin films (plated wire) seems to support this contention. Batch fabricated memory devices are urgently needed if larger memories at lower costs are to be achieved. Manufacturers are fully aware of this situation and are striving to meet the demand. Workable, low cost devices will be developed as more knowledge is gained concerning manufacturing techniques. It is also to be assumed that the electronics for such devices (drivers, sense amplifiers, etc) will correspondingly be reduced in size

through integrated circuit techniques so that the entire memory will be compatible with that which is required by the proposed system.

3. MEMORY CONFIGURATION

Present day memory technology dictates that "memory size conflicts with memory speed and that compromises have to be made between pulse rise time and transmission line attenuation, between signal amplitude and drive power independent of the technology used".⁸ Certain fundamental barriers exist to the optimization of speed, capacity, and cost which can be defined in terms of device switching speed, circuit delay due to gain-bandwidth of semi-conductor circuits, and transmission delay in the array itself.

There exists a traditional hierarchy of memory types in which ferrite cores are confined to main memory. Although other technologies offer future promise for increased speed, large capacity and low cost, "magnitude greater strides are necessary to make an impact on large capacity memories".⁵ It is therefore reasonable to confine our investigation to a system which uses ferrite cores as the storage medium in central memory.

It has been stated that data transfer rates of the magnitude envisioned for our hypothetical computer will be achieved only through the efficient distribution of data from a central memory or between processing elements. By "efficient" it is meant that memory and processing elements be kept active at all times. That is to say, a sufficient flow of data and instruction should be maintained so that "waiting-time" is reduced to a negligible amount.

By and large, the greater portion of "waiting-time" in any computer program is spent in memory fetches and stores. That this is true is evidenced by the fact that memory access times are in the order of a magnitude(s) larger than say add times. Consequently, the greater portion of computation time is spent in getting data into and out of the memory.

Two approaches to increasing data flow rates from memory are: 1) develop faster memories and 2) interleave memories.⁹ With respect to the former, it has been pointed out that there are fundamental barriers to increasing speed and capacity and lowering costs.

Considerable progress has been made in core memory speed capacity since 1963. This is shown in Figure 1. It can be seen that Rajchman underestimated memory

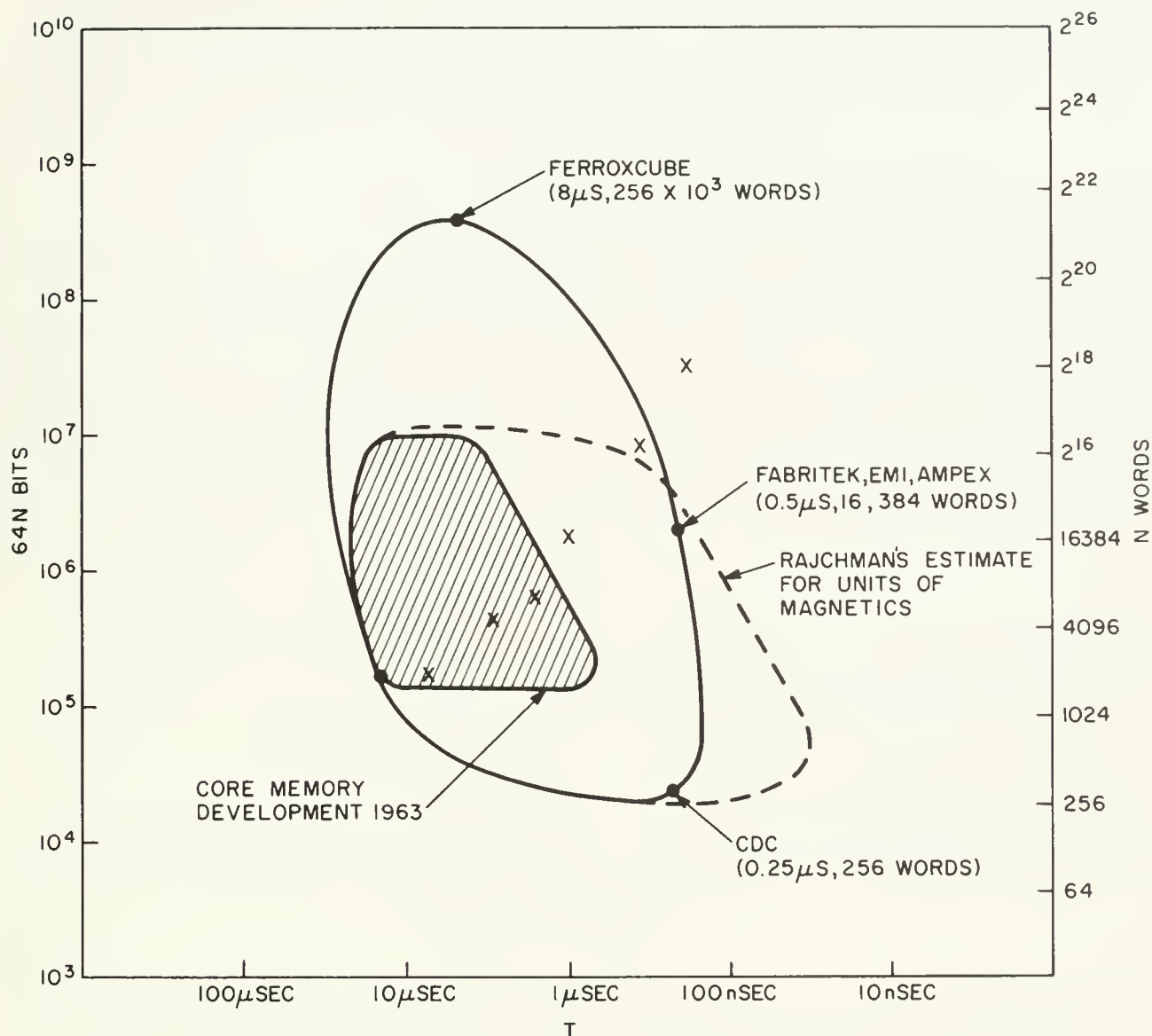


Figure 1. Limitations of Present-Day Core Memories

capacity for slower memories. However, the faster memories are still within his estimated boundaries and so may be used as a convenient guideline for future development. It is to be noted that the faster memories show a marked decline in their storage capacity. Therefore, for any given memory module size, the faster memories would require more modules. Apparently a trade-off must be made between speed and module capacity.

It will be recalled that the desired memory capacity of the proposed computer is 16×10^6 words and that data transfer rates should be in the order of 10^9 words per second. To accomplish such a rate with a single bank of memory, a cycle time of one nanosecond must be obtained. It is obvious that such memory speeds do not exist.

Therefore, in order to affect such a cycle time, many words must be extracted simultaneously. For instance, 1000 words extracted in one microsecond yields an effective data transfer rate of 10^9 words per second. In general, the number of words to be extracted simultaneously can be determined by the following relation:

$$N = 10^9 t$$

where N = number of words and t = cycle time.

If it is assumed that only one word may be extracted per bank per cycle time, then for a one microsecond cycle time, 1000 banks of memory would be required. If an array is formed and so organized that successive requests always address inactive banks of memory, then more banks are required but they may have longer cycle times. Therefore, for a given request rate (in this case 10^9) and memory cycle time, the size of the array may be determined by:

$$A^2 = Rt$$

where A^2 = total number of memory banks in the array. Since we have specified that 16×10^6 words would be contained in the memory,

$$aA^2 = 16 \times 10^6$$

where a = number of words per bank, and A^2 = number of memory banks therefore,

$$A^2 = \frac{16 \times 10^6}{a} = 10^9 t$$

We may now relate the module size (number of words per bank) and memory speed. Thus $at = 16 \times 10^{-3}$. Interpreting this for different memory speeds yields Table 1.

TABLE 1
SIZE-SPEED RELATIONS FOR 16×10^6 WORDS
OPERATING AT 10^9 WORDS / SEC

MEMORY CYCLE TIME (sec)(t)	WORDS / MODULE (a)	ARRAY SIZE (A)
0.25×10^{-6}	64×10^3	15.8×15.8
0.5×10^{-6}	32×10^3	22.4×22.4
1.0×10^{-6}	16×10^3	31.5×31.5
2.0×10^{-6}	8×10^3	44.8×44.8
4.0×10^{-6}	4×10^3	63×63
8.0×10^{-6}	2×10^3	89×89

A plot of points corresponding to the cycle time/capacity relations of Table 1, when superimposed on the estimates of Figure 1, reveals that the required size of the the $0.25\mu s$ memory falls outside the bounds for present day memories as well as the predicted limits. Therefore, this memory is inapplicable to our system.

Similarly, the $0.5\mu s$ memory and the $8.0\mu s$ memory should be discounted since their capacities are almost at the limits of what is being produced. This narrows the choice of memory modules to three; $1.0\mu s/16 \times 10^3$ words, $2.0\mu s/8 \times 10^3$ words, or $4.0\mu s/4 \times 10^3$ words.

From an economic standpoint, it can be seen from Figure 2 that the slowest memory ($4\mu s$) would probably cost less than the faster memories even though more modules would be needed. Therefore, the $4.0\mu s$ memory should be favored providing it is compatible with the remaining specifications of the proposed system.

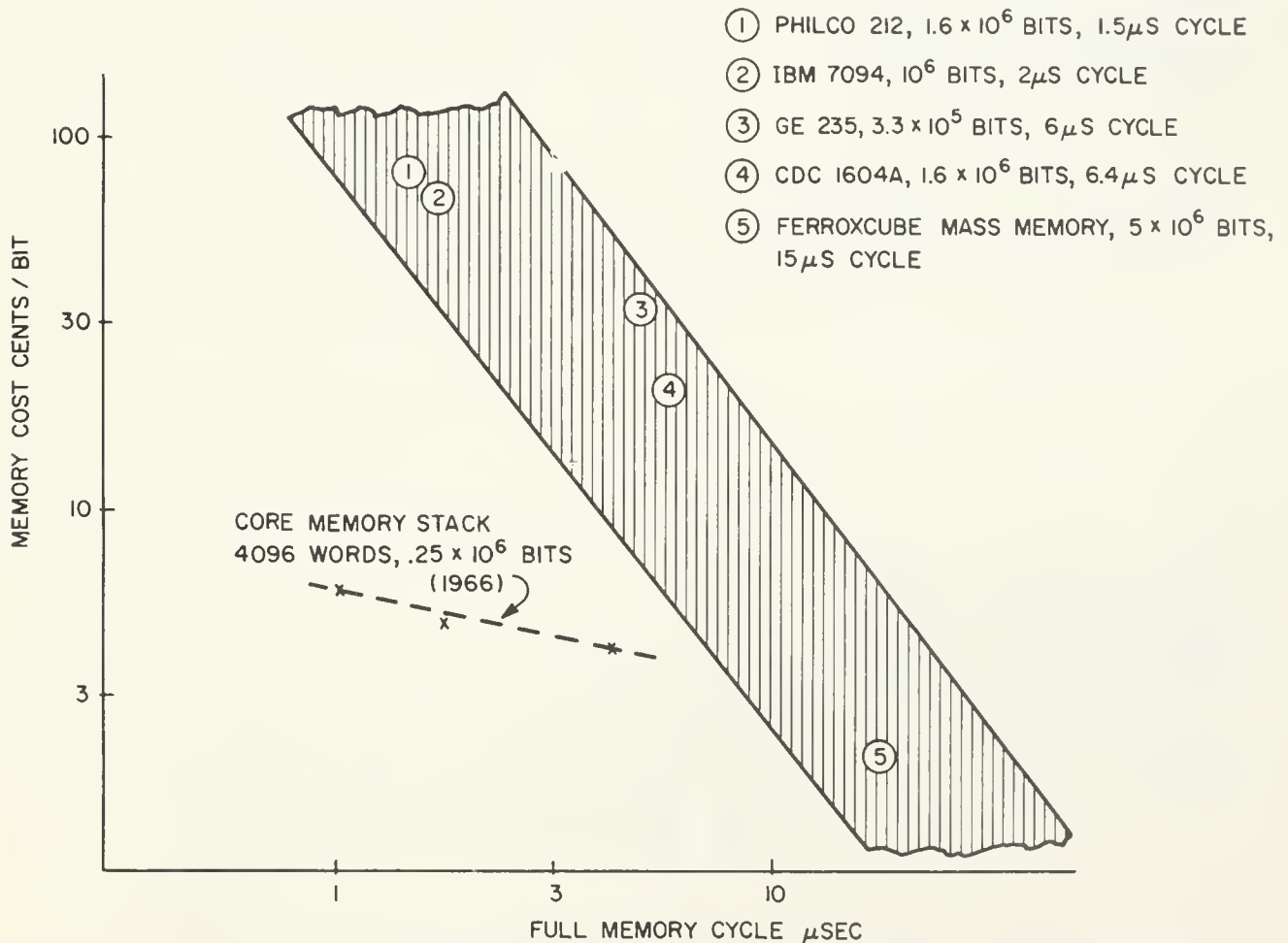


Figure 2. Random Access Memory Cost, 1964

The memory configuration would be an array of 64x64 banks, each containing 4096 words, (each word containing 64 bits), and operating with a $4.0\mu\text{s}$ cycle time. This is shown in Figure 3.

It has been suggested that in average coding, CPU's may make memory references every 60 nanoseconds.⁴ If we consider one column of the memory array as a memory system, then by interleaving outputs from successive banks onto a common line and addressing the memory in successive banks, we can continuously extract 64 words/column every $4.0\mu\text{s}$. This is an effective memory cycle time of 62.5 nanoseconds. Hence, the memory is compatible with the CPU "turn-around" time. Since there are 64 columns of memory, 4096 words may be extracted in $4.0\mu\text{s}$. This affects a data transfer rate of 1.02×10^9 words/sec.

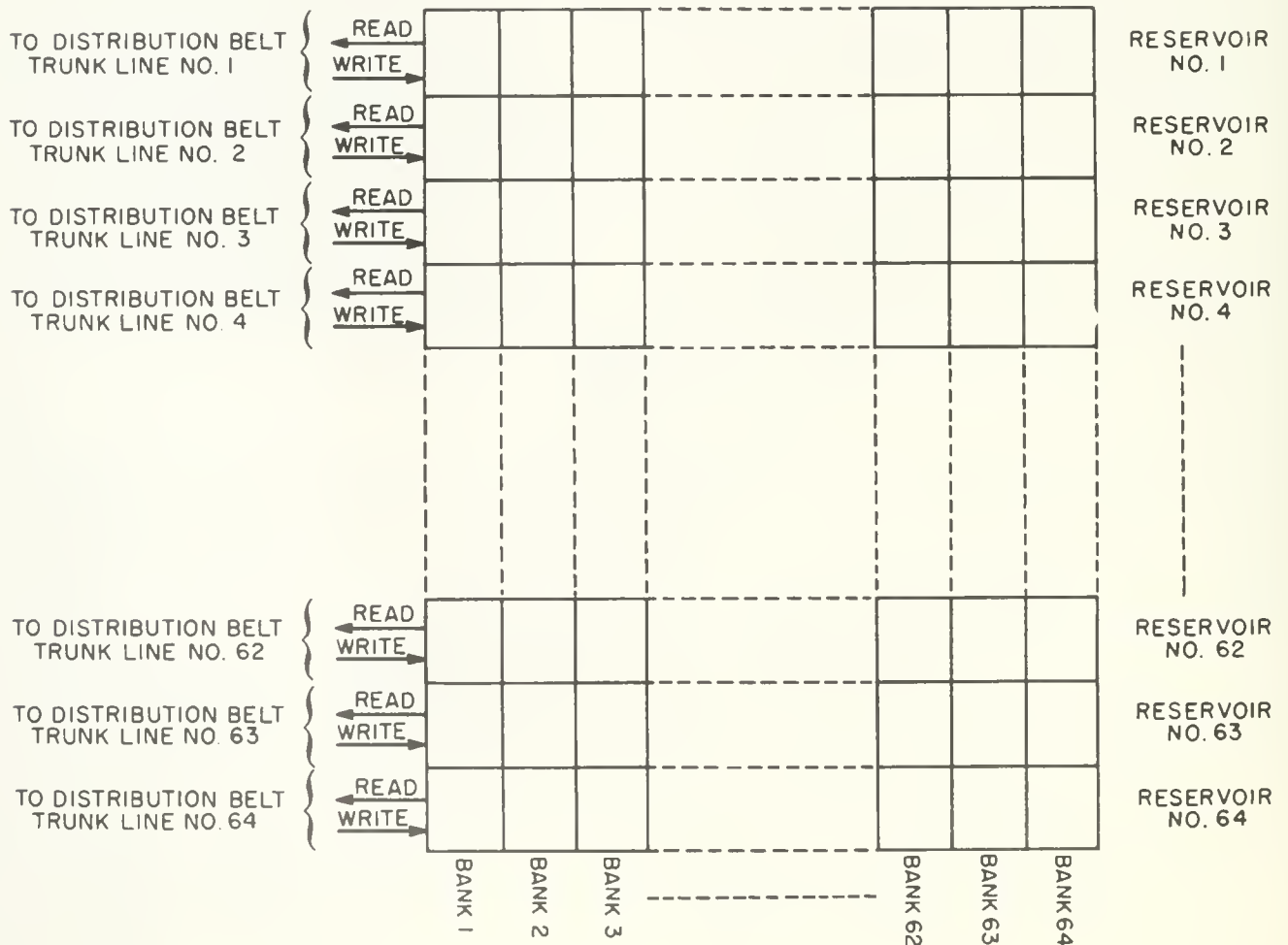


Figure 3. Central Memory (64 Reservoirs, 64 Banks/Reservoir, 4096 Words/Bank, 64 Bits/Word)

4. MEMORY SELECTION

From the foregoing expression relating the number of words to be extracted from the memory and the memory cycle time ($N = 10^9 t$), and if we assume our memory to have an effective cycle time of 62.5 nanoseconds, then to maintain a data flow rate of 10^9 words per second 62.5 words should be extracted simultaneously. Since this is not a convenient number of words, 64 words will be extracted simultaneously.

In order to locate a particular word in memory, its column, row, and word number must be identified. Since there are 64 columns, 6 binary bits will serve to locate the column. Similarly, 6 bits may serve for the row. Since each bank of memory, now identified by a column and row, contains 4096 words, 12 bits will be needed to locate the word. Consequently, 24 binary bits will be necessary to locate a particular word in memory. The remaining 40 bits of the word may be used for the Op-code, destination tag, indirect address tag, etc. Also to be considered is the practicality of two memory addresses per word, the second address being the next word to be extracted. This may not speed up the transfer rate, but may allow longer "turn-around" times so that the CPU may spend more time "computing". How this might be accomplished from a hardware standpoint will not be discussed in this report.

It has been shown that if a data transfer rate of 10^9 words/sec is to be obtained, only inactive banks of memory may be addressed. Simultaneous or successive requests for the same bank of memory would reduce the transfer rate considerably due to the limitation of memory cycle time.

Circumvention of conflicts may be achieved in three ways: 1) control of memory fetches through software so as to avoid conflicts, 2) control of memory fetches by an executive processor which will keep track of active memory locations, and 3) built in hardware which will automatically resolve conflicts.

Control of memory fetches through software is probably the ideal way of preventing conflicts. However, this method would place the burden of bookkeeping upon the programmer and with the complexity of the program may become a formidable task. Such being the case, the programmer may elect to assign sections of memory to specific CPU's. This would preclude free access to any word in memory by any CPU thus reducing the computing power of the system. Effectively, this would be no different than using multiple computers.

The use of an executive processor to keep track of active memory banks removes the burden of bookkeeping from the programmer, but entails the use of a fairly large size memory within the processor to retain conflicting memory requests. Since only one word may be extracted from a given memory within $4.0\mu s$, and since memory requests from a single CPU may be made at 60 nanosecond intervals, if successive requests are made for the same memory bank, then 65 requests must be retained for future issuance. When we consider that there are 64 CPU's issuing requests at 60 nanosecond intervals and each CPU may access the same memory bank, then it is evident that 4095 requests must be stored every $4.0\mu s$. Hence, the executive processor may require a memory almost as large as CM itself.

Built-in hardware which will automatically resolve conflicts is an alternative but can result in increased costs and reduction of data transfer rate. However, this may be the only practical solution to the problem.

A comment concerning the specification for 60 nanosecond "turn-around" time should be made. Current philosophy of programming with regard to memory fetches is one of "request-receive-request". That is to say, data from memory is requested by the processing unit, acted upon, and then more data is requested. If it is assumed that such a philosophy is to be maintained for parallel processing techniques, then CPU's need not have a 60 nanosecond "turn-around" time.

The access time (A_t) of a memory system is that interval between initiating the request and receiving a usable output from the memory. This time interval includes address decoding time (t_A), core switching time (t_S), signal propagation transmission time (t_t), and sense amplifier delay time (t_D). Thus,

$$A_t = t_A + t_S + t_t + t_D .$$

In a word organized memory, a relatively large decoding switch is necessary to translate the n -bit binary address into the selection of one word among $N = 2^n$ words. This decoding is usually carried out in a number of steps usually less than n .⁵ Each of these steps requires hardware to carry out. If we assume that a 6-bit decode may be accomplished using three steps (flip-flop storage, fan-out, decode), and if each hardware gate has a propagation delay of 5 nanoseconds, then the address decode time can be 20 nanoseconds. (Two delays are needed for the flip-flop.)

The switching time of a ferrite core is dependent upon the magnitude and rise time of the driving current pulse. Although past predictions concerning fundamental

limits have been outdated by technical progress, there still remains a finite time for cores to switch. The most optimistic predictions indicate that switching times of ferrite cores will not be reduced below 50 nanoseconds.⁸

The delay time along the sense windings is primarily due to the inductance and capacitance of the line and is proportional to the number of words and the spacing between elements. By using small cores and employing antisymmetrical drive with center-of-the-line sensing,¹⁰ it may be possible to achieve 20 or 30 nanosecond delays.

By virtue of the nature of sensing amplifiers, a delay is encountered between input and output which is proportional to the number of amplifying stages. Even with the use of integrated circuitry, this delay cannot be eliminated. It is estimated that sense amplifier delay may be in the order of 20 nanoseconds.

From the foregoing, it can be seen that minimum access times will be in the order of 110-120 nanoseconds. Thus, if a "request-receive-request" technique is employed and if an instruction execution time is in the order of say 10 nanoseconds, then the minimum required memory "turn-around" time for any CPU is in the order of 150 nanoseconds. It is therefore evident that the specified "turn-around" time demands independence of memory request source and destination.

5. MEMORY REQUEST CONFLICTS

If memory request conflicts are not to be eliminated through software or executive processor techniques, then hardware must resolve them. That such conflicts will occur can be surmised from the probability that they will occur. Thus, the probability that two CPU's will address the same memory bank if 64 requests are made simultaneously can be estimated by:

$$\begin{aligned} \text{Probability of conflict between 2 CPU's} &= \frac{64!}{2!62!} \times \frac{1}{64^2} \\ &= \frac{64 \times 63}{2 \times 64 \times 64} = \frac{63}{128} = 0.494 . \end{aligned}$$

This of course is too high a probability to be ignored. The probability of conflict between two or more CPU's can be estimated from:

$$\text{Probability of conflict} = \sum_{p=2}^{n-1} \frac{n!}{p!(n-p)!n^p}$$

It can be seen that this expression yields a very high probability of conflict.

By and large, three categories of conflicts can occur. In the first, two or more banks of memory in a common column are being addressed. These requests must be separated due to the handling capabilities of the column read-out lines (trunk lines). A second category conflict is one in which two or more simultaneous requests are made for information contained in a common bank of memory (same bank, different words). These requests must be separated because of the memory cycle time. In category three falls those conflicts arising from successive requests for information from a common bank occurring in an interval of time less than the memory cycle time. In this respect, they are similar to category two conflicts but must be resolved differently.

6. CONFLICT CHECKER

The number of conflict checks that must necessarily be made is equal to the number of combinations that can be formed from the total number of simultaneous requests taken two at a time. Since the address format permits separate selection of column, row, and word, each of these groupings may be checked separately to determine if conflicts exists.

If all 64 processors are permitted to address CM simultaneously, the number of combinations per column that must be checked for conflict is:

$$\frac{64!}{2!62!} = \frac{64 \times 63}{2} = 32 \times 63 = 1890 + 126 = 2016 .$$

Assuming that each of these is a two-input gate, and since each CPU can address any one of 64 columns, then the total number of gates needed is:

$$64 \times 2016 = 120960 + 8064 = 129,024 .$$

Fewer gates are needed if fewer simultaneous memory requests are allowed. In general, the total number of gates that would be needed to check for conflicts in any one of 64 columns is:

$$\text{Total number of gates} = 64 \left[nC_2 + \frac{64}{n} C_2 \right]$$

where n = number of simultaneous memory requests allowed. Thus, if 16 requests were allowed:

$$\begin{aligned} \text{Total number of gates} &= 64 \left[16C_2 + \frac{64}{16} C_2 \right] = 64 \left[\frac{16 \times 15}{2} + \frac{4 \times 3}{2} \right] \\ &= 64 [120 + 6] = 64 \times 126 = 8064 . \end{aligned}$$

A plot of total number of gates vs. number of simultaneous requests is shown in Figure 4.

A further advantage to be gained by allowing fewer simultaneous requests is a reduction in the probability of conflict. Thus, for 16 requests, the probability of conflict between two requests is reduced to 0.029. This implies fewer delays due to conflicts.

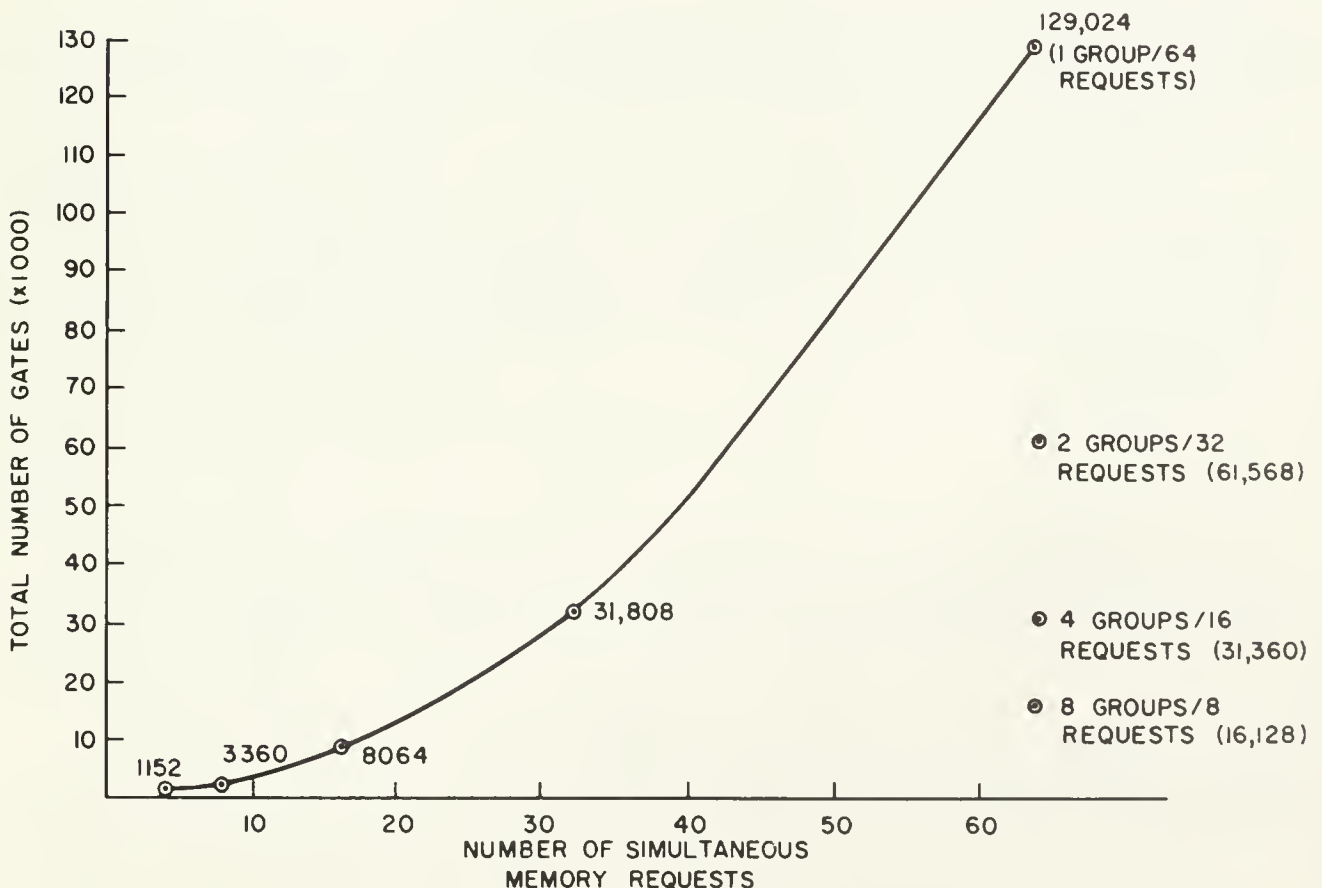


Figure 4. Total Number of Gates vs Number of Simultaneous Requests for Conflict Checker

Similar reductions in the number of gates are obtained if checks are made by groups. For instance, if the 64 requests are divided into four groups of 16 requests each, then each group requires $16C_2 = 120$ gates. Therefore, 480 gates will be required to determine whether conflicts exist within each grouping. In addition, four 16-input "or" gates plus six 2-input "and" gates will be necessary to complete the check. This yields 490 gates per column or a total of 31,360 gates as compared to 129,024 gates without grouping. However, regardless of the way 64 requests are grouped for checking purposes, the probability of conflict will remain unchanged.

If the number of simultaneous requests is reduced, a "time-sharing" scheme may be employed to maintain the required "turn-around" time. Thus, 64 simultaneous requests at 60 nanosecond intervals permits a CPU "turn-around" time of 60 nanoseconds. Similarly, 32 simultaneous requests at 30 nanosecond intervals maintains the CPU "turn-around" time. In general, the following relation holds true:

$$N = \frac{64 \tau}{60 \times 10^{-9}} = 1.07 \times 10^9 \tau$$

where N = number of simultaneous memory requests, and τ = interval between requests.

Ideally, for a 64x64 matrix of memory banks, $64 \times \tau$ should equal the memory cycle time so that there is minimum latency time in data flow. Thus, $64 \tau = T$ where T = memory cycle time. Relating cycle time and number of simultaneous requests for 60 nanosecond "turn-around" time yields:

$$N = \frac{T}{60 \times 10^{-9}}$$

As has been shown, it is advantageous to reduce the number of simultaneous requests both from a probability standpoint and from a hardware standpoint. It has also been shown that to do so would require a reduction in the memory cycle time if the CPU "turn-around" time is to be maintained. It may be argued that the faster memories would increase the cost. However, the increased memory cost would be offset by the reduction in cost of the hardware necessary for the conflict checker.

A re-examination of Figure 1 reveals that the memories previously chosen for consideration are still applicable. However, the faster memory should now be chosen so as to secure the greatest reduction in the number of gates required for the conflict

checker. Thus, choosing the $1\mu\text{s}$ memory in a 64×64 array with 4096 words per bank allows 16 simultaneous requests for memory at 15 nanosecond intervals.

7. MEMORY ADDRESSING - THE TIME-SHARING SYSTEM

The "time-sharing" scheme for memory addressing is illustrated in Figures 5A and 5B. Thirty-two CPU's are permitted to enter their requests simultaneously onto a "belt" network. Sixteen of these requests are processed during the first "slot" interval while the remaining sixteen requests are moving along the belt. These requests are processed in the second interval. Thirty-two more requests are then entered onto the belt with sixteen requests being processed in each of the third and fourth intervals. The sequence then repeats with the original thirty-two CPU's issuing new requests providing no conflicts have occurred. In this manner, every processor is permitted to make memory requests at 60 nanosecond intervals.

This "belt" configuration lends itself to conform with physical distances which may occur between processing units. Processing units associated with each group are separated by a 5-nanosecond delay line which may be realized with a co-ax cable approximately 3.5 feet long. (RG62/U co-ax cable has a propagation delay of 1.2 to 1.5 nanoseconds per foot.)

It is to be noted that the "belt" is continuous. At the same time that the address bits are sent to their respective decodes, they start a journey through a delay line with their destination being one stage further up the belt. This journey will take a minimum of 60 nanoseconds. However, this journey will be prematurely terminated if no conflict exist. Conversely, should a conflict occur, each CPU involved in the conflict moves up one stage of "priority" and gets a chance to reissue its request at 60 nanosecond intervals. Although priority has been arbitrarily established in favor of CPU's associated with the upper stages of the belt, these CPU's cannot issue new requests until conflicts have been resolved.

Each half of each stage of the belt is identical, employing one inverter, two three-input "nand" circuits, and one three-input "nor" circuit. Such a functional block is shown in Figure 6 and requires 10 pins and 14 base-emitter junctions. These requirements are completely compatible with present-day integrated circuit devices and therefore should be realizable in integrated circuit form. Two modules per stage per bit would be required and since there are sixteen stages and twenty-four bits, 768 modules

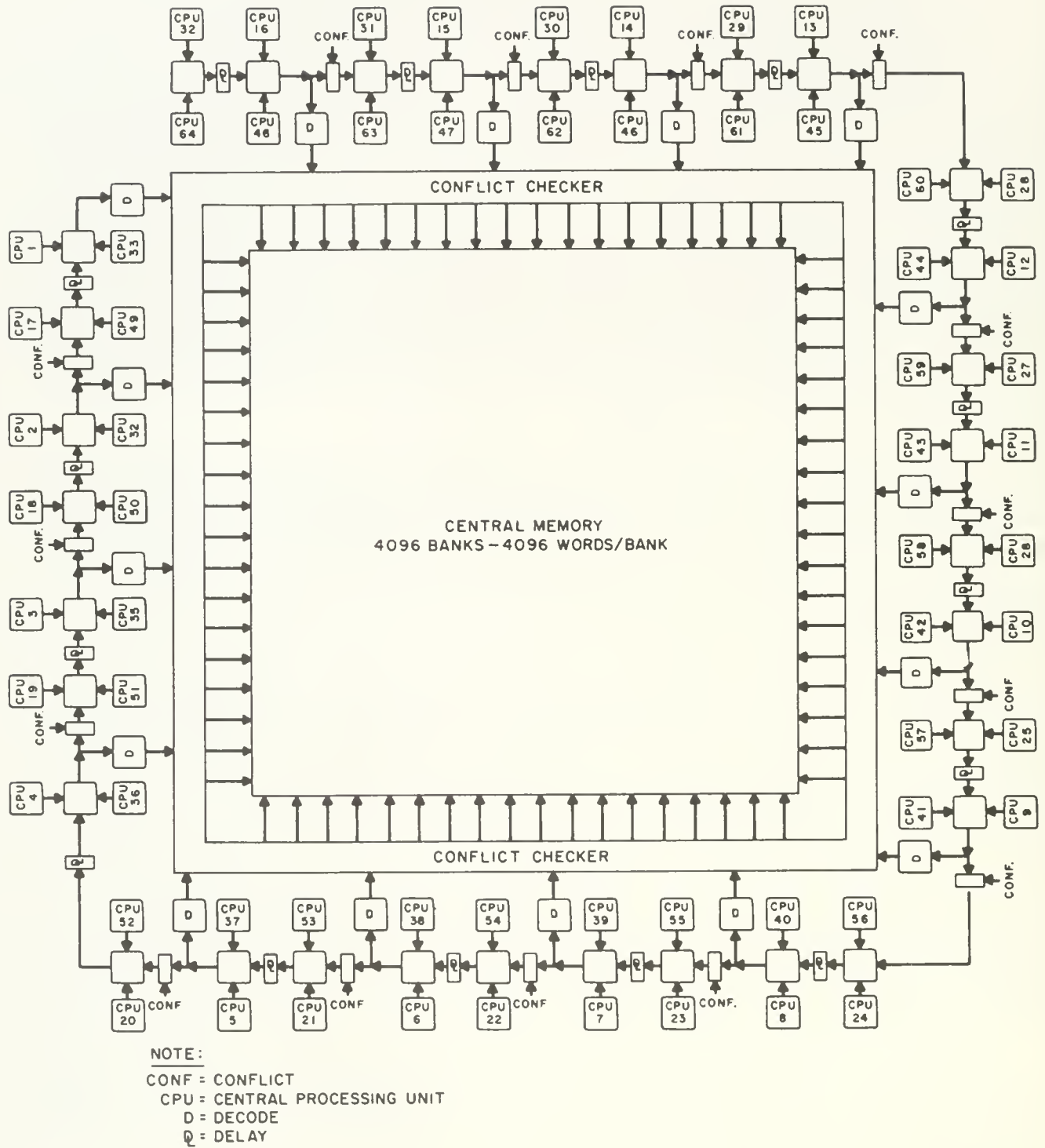


Figure 5A. Memory Address Selection Block Diagram

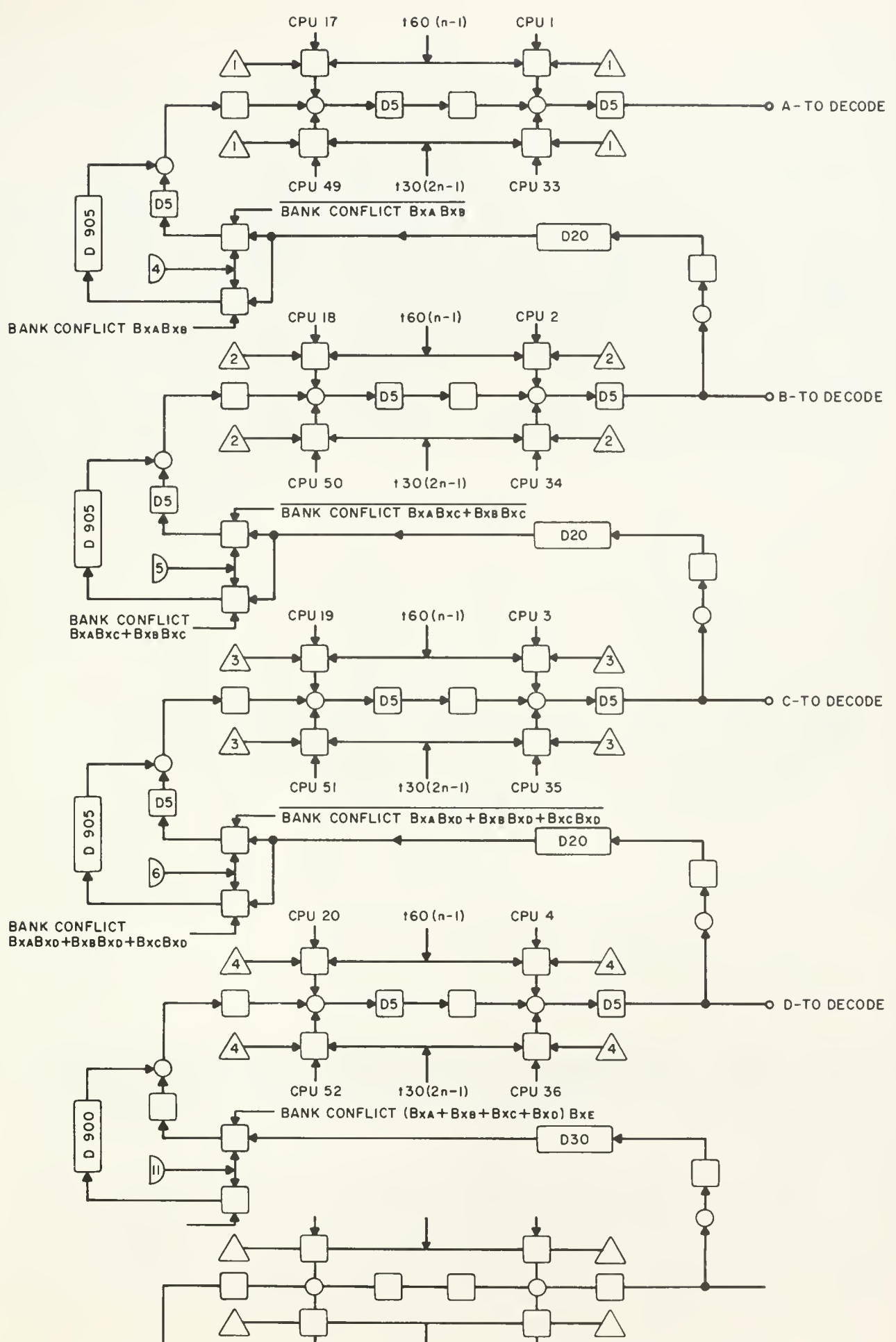


Figure 5B. Memory Addressing "Time-Sharing" Systems (1-Bit)

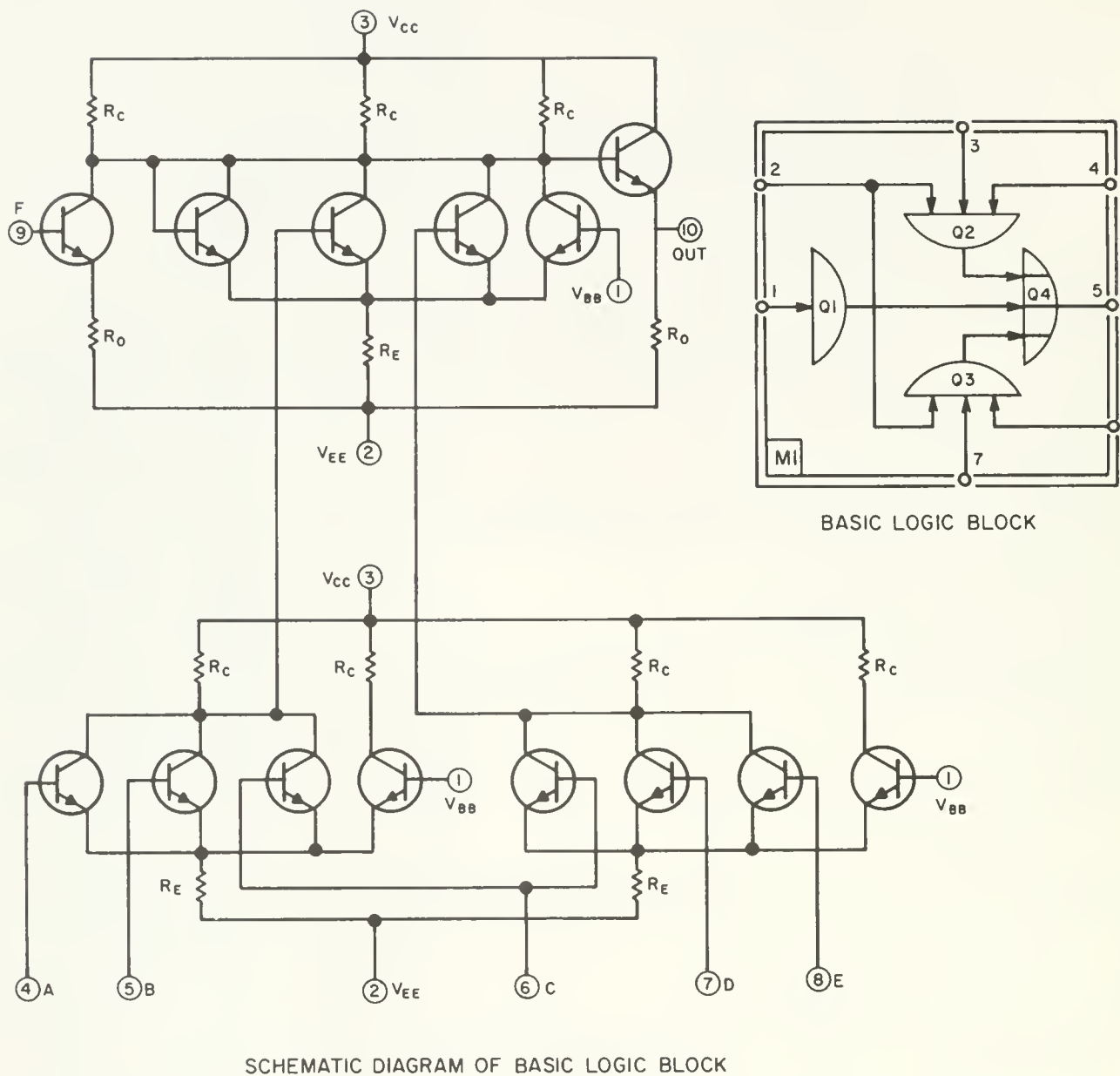


Figure 6. Basic Address Time-Sharing Logic Block

would be needed. At present-day prices for integrated circuitry (\$0.50 per transistor circuit), this module should cost \$7.00. Therefore, total cost for these modules would be \$5376.

It is essential that each module have propagation delays which are maintained within fairly narrow boundaries and that these delays are uniform from module to module. Use of integrated circuitry for this module should produce uniformity of parameters within the module and insure consistency of propagation delays from module to module.

The use of emitter-follower current-switching logic should provide for fairly large fan-outs and constant power requirements.

Different categories of conflict requires that reissuance of memory requests be delayed different intervals of time. Therefore, two separate paths are provided on the belt. The short path provides a delay of 60 nanoseconds and is applicable where a conflict exists because of trunk line handling capability. This condition is fostered by two requests for the same column but different rows. The long path provides a delay of $1\mu s$ and is applicable where two requests are made for information contained in a common memory bank. The selection of the proper path is controlled by the conflict checker and may be accomplished by the network of Figure 7. Such a module may be fabricated in integrated form at an estimated cost of \$5.00. Fifteen modules per bit would be needed at a total cost of \$1,800.

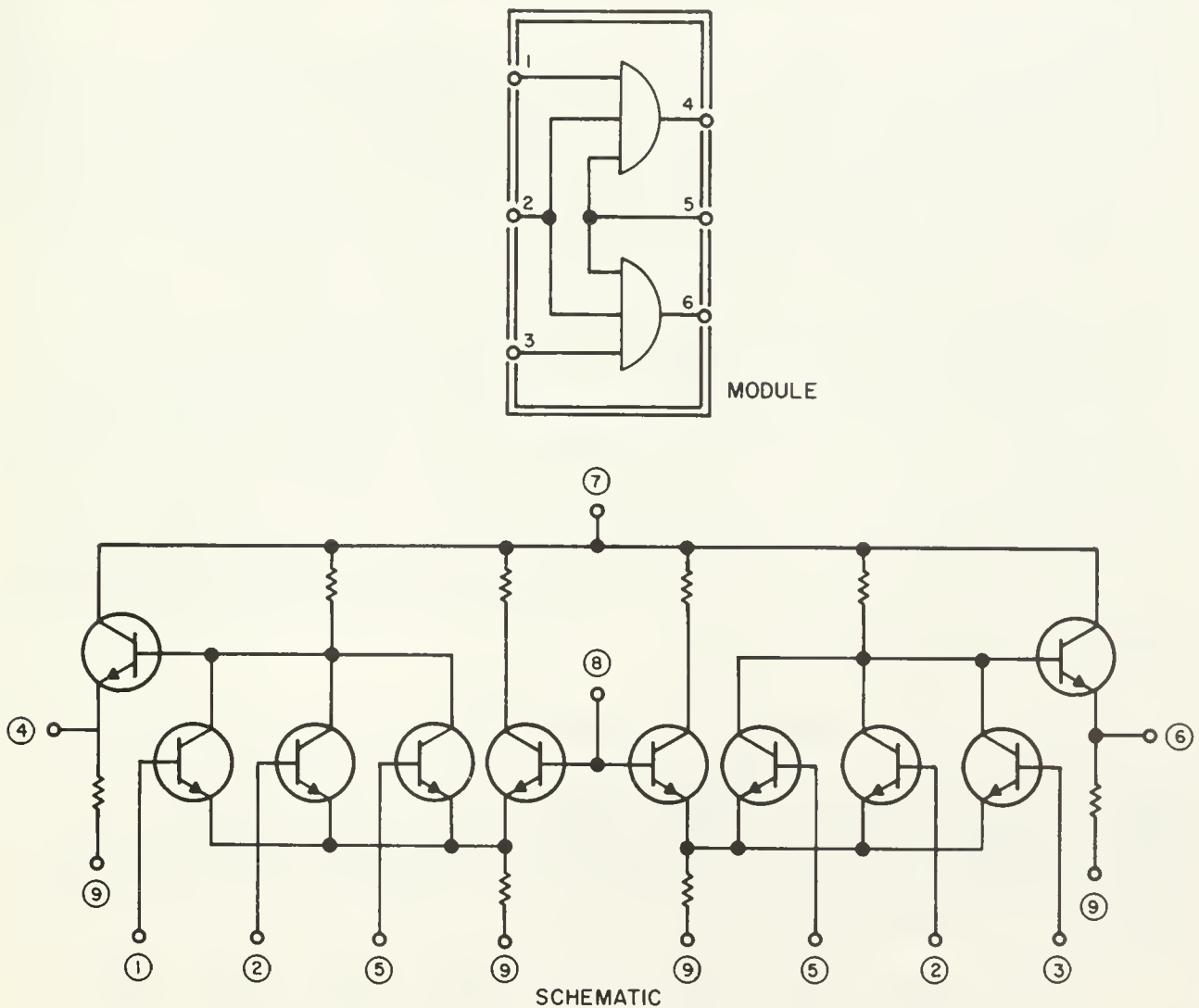


Figure 7. Long and Short Path Switching Network

8. THE CONFLICT CHECKER

As has been shown previously, with 16 requests for memory being made simultaneously, 120 checks per column would necessarily be made to determine if any two requests are in conflict. This would normally require 120 two-input gates per column. However, fewer gates would be needed if conflict checks are made in groups. Thus 2 groups of 8 requests would require $2 \times 8C_2 + 2C_2 = 2 \times \frac{8 \times 7}{2} + 1 = 57$ gates; 4 groups of 4 requests would require $4 \times 4C_2 + 4C_2 = 4 \times \frac{4 \times 3}{2} + \frac{4 \times 3}{2} = 30$ gates. It is obvious that fewer gates are required if 4 groups of 4 requests are used for checking purposes.

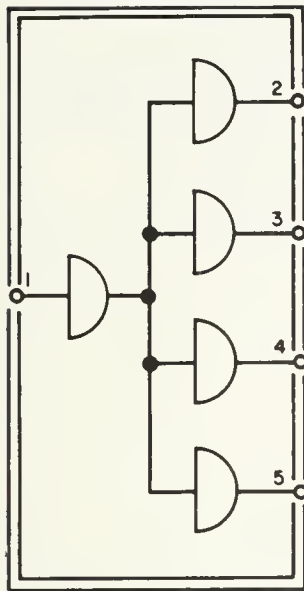
A close examination of the nature of conflicts reveals that no word conflict can occur without a bank conflict, and no bank conflict can occur without a trunk (column) conflict. Therefore, the sequence in determining conflicts should be trunk, bank, and word.

The output of each stage of the address selection "belt" is applied to a decoding network. Trunk, bank, and word selections are decoded separately. The six-bit trunk selection decodes into 64 outputs ($2^6 = 64$). The decoding mechanism is a six-input "nand" gate. Each bit of the trunk address must be applied to 32 gates. If the fan-out capabilities of the driving modules is limited to five, then a minimum of seven drivers per bit would normally be needed. However, only four drivers per bit would be needed if "current-sharing" techniques are employed.¹¹ The driver fan-out module is shown in Figure 8 and may be fabricated in integrated circuit form for \$3.00 per module. Since 24 bits would be used per address decode, twenty-four such modules would be used at a total cost of \$72.

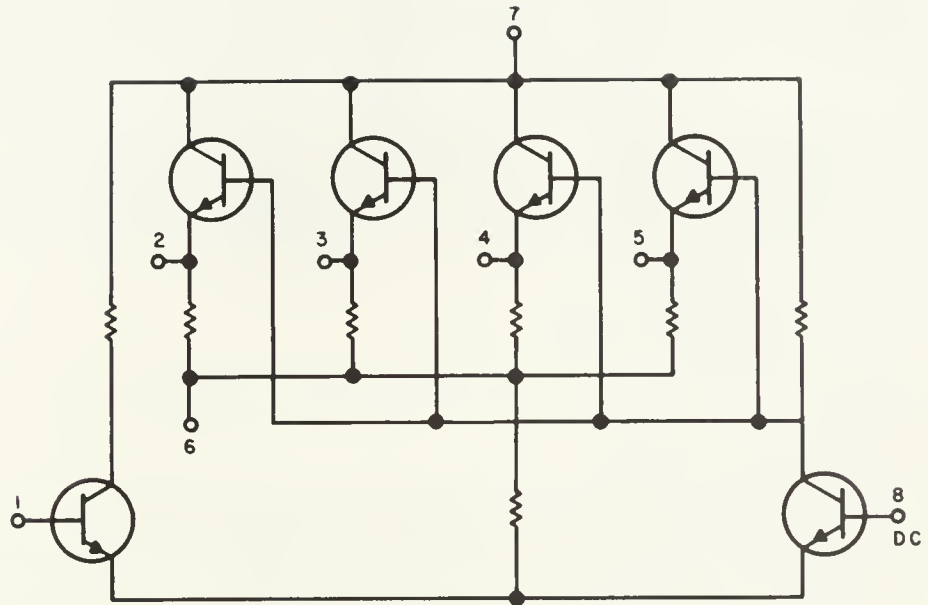
The total number of decoding gates needed is $64 \times 16 \times 4 = 4096$. If each six-input "nand" gate may be fabricated in integrated circuit form for \$4.00, then the cost of the decoding mechanism would be \$16,384.

Checking for trunk conflicts in groups of four may be accomplished in seven gates per group per trunk. Although there are eleven combinations that will produce conflicts ($4C_2 + 4C_3 + 4C_4 = \frac{4 \times 3}{2} + \frac{4 \times 3 \times 2}{3 \times 2} + \frac{4 \times 3 \times 2}{4 \times 3 \times 2} = 6 + 4 + 1 = 11$), only combinations of two need be checked. That this is so is evidenced by a reduction of the Boolean expression for the eleven combinations.

$$\begin{aligned} &ABCD + \bar{A}BCD + A\bar{B}CD + AB\bar{C}D + ABC\bar{D} + A\bar{B}C\bar{D} + \bar{A}BC\bar{D} + \bar{A}\bar{B}CD + A\bar{B}\bar{C}D \\ &+ A\bar{B}C\bar{D} + \bar{A}\bar{B}\bar{C}D = AB + AC + AD + BC + BD + CD . \end{aligned}$$



FAN-OUT MODULE



FAN-OUT MODULE SCHEMATIC

Figure 8. Driver Fan-Out Module

The trunk conflict checker is shown in Figure 9, can be integrated and should cost approximately \$18 per module. Five modules per trunk would be needed, and a similar amount for each bank. Therefore, $18 \times 5 \times 64 \times 2 = \$11,520$ would be the cost for the conflict checker. No conflict checks are made for word selection since the occurrence of both trunk and bank conflicts would be sufficient to determine that two requests have been made for a common bank of memory.

9. TRUNK-BANK SELECTION

Having detected that a trunk conflict exists, it remains to be determined which trunk-bank combination will be selected. In effect, a priority must be established. The addressing "belt" priority has arbitrarily been established so that the uppermost stage (Decode A of Figure 5) has preference in cases of conflict.

It will be noted that sixteen memory requests are made simultaneously and that these requests are divided into four groups of four requests each. Therefore, priority

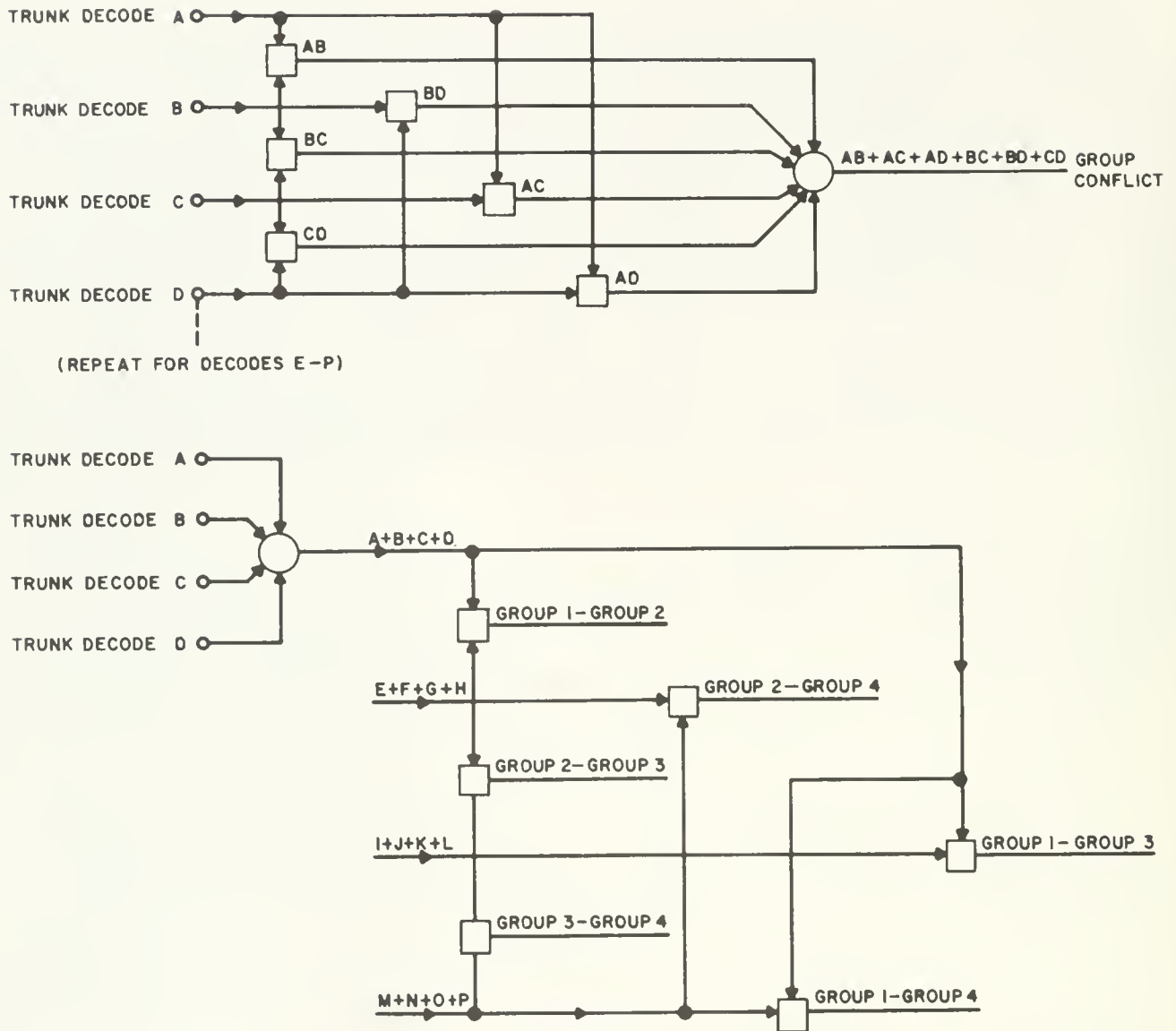


Figure 9. Trunk Conflict Checker

for each group may be established by the Boolean expression $A + \bar{A}B + \bar{A}\bar{B}C + \bar{A}\bar{B}\bar{C}D$. This expression is one of exclusivity and establishes that one and only one decode will be recognized.

The priority system requires that each trunk decode be combined with its corresponding bank decode in such a manner that one and only one trunk-bank combination be selected for each trunk selected. This implies that satisfaction be obtained for the Boolean expression:

$$TxBx = TxA Bx_a + \bar{T}x_a Txb Bxb + \bar{T}x_a \bar{T}xb Txc Bxc + \bar{T}x_a \bar{T}xb \bar{T}xc Txd Bxd .$$

The expression is satisfied by the network of Figure 10. This scheme requires that every trunk priority signal be gated with every bank decode. Consequently, 4096 such networks must be fabricated. The trunk-bank selection module may be fabricated in integrated circuit form for approximately \$6.00. Since 4096 modules are needed, the total cost for these modules would be \$24,576. In addition, 64 fan-out modules per trunk would be needed for a total of $2 \times 32 \times 64 = 4096$ modules. Therefore, an additional cost of \$12,288 will be incurred.

The scheme for selection of the proper word within the selected bank is not taken up in this report. It is felt that such a scheme is dependent on the organization of the memory bank (i.e. word organized, bit organized, etc.).

10. DATA DISTRIBUTION

The basic scheme for information distribution consists of a series of "conveyer belts", one belt for each bit of the data (information) word and one belt for each bit of the operand. Hence, a total of 90 belts would be used. Each belt is essentially a delay line of 64 stages (one stage for each CPU) with provision for entering and exiting at each stage when appropriate. Data enters the belt from CM by means of a "trunk" line consisting of 64 cables. Each trunk line is assigned a specific entry point on the belt. Trunk lines carry data from specified CM reservoirs. Since there are to be 64 reservoirs, 64 trunk lines would be used, each entering a different stage of the belt. Likewise, each CPU may enter data on the belt at specified points. Exit from the belt may be made from each stage and may be directed either to CM or to a CPU. Data travelling on the belt moves at the rate of 15 nanoseconds per stage. Therefore, to traverse the entire belt would require approximately 1 microsecond. (This is the cycle time of a memory bank.) Data, once entered on the belt, must make one complete trip around the belt and is then removed from the belt. This insures that each CPU has access at 15 nanosecond intervals to any information placed on the belt. Thus, the longest that any CPU must wait for requested data would be 1 microsecond from the time that the data is placed on the belt. Conversely, the shortest wait is 45 nanoseconds. (See Figure 11.)

At the same time that data is entered on the "data belt", the operand is entered on the "address belt". In effect, each word carries with it its own address thus identifying itself. The address so entered is checked by each CPU for equivalence of address, thus identifying the information it requested. The information is then routed to either the CPU (fetch) or to CM (store). (See Figure 12.)

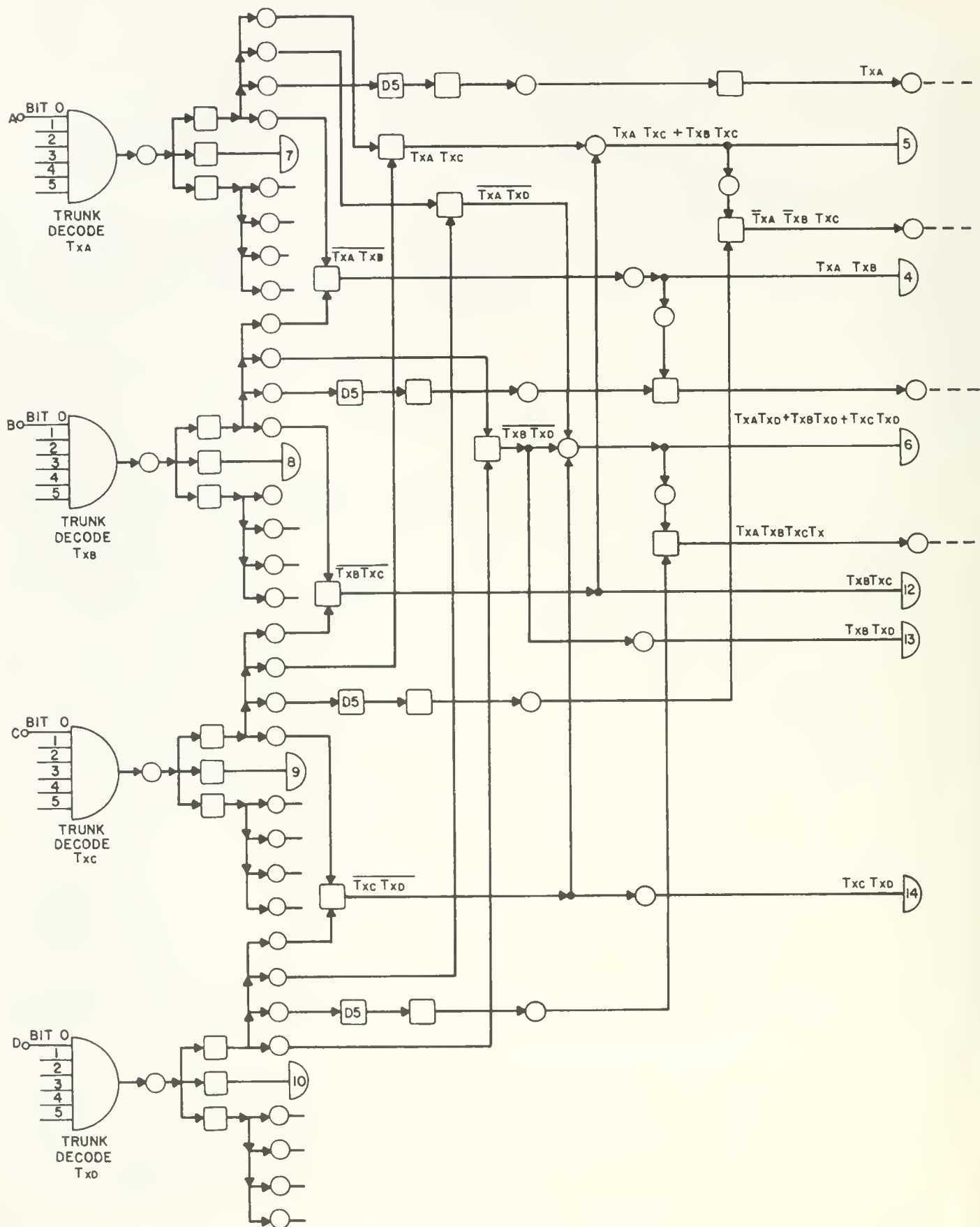


Figure 10. Trunk-Bank Selection (Sheet 1)

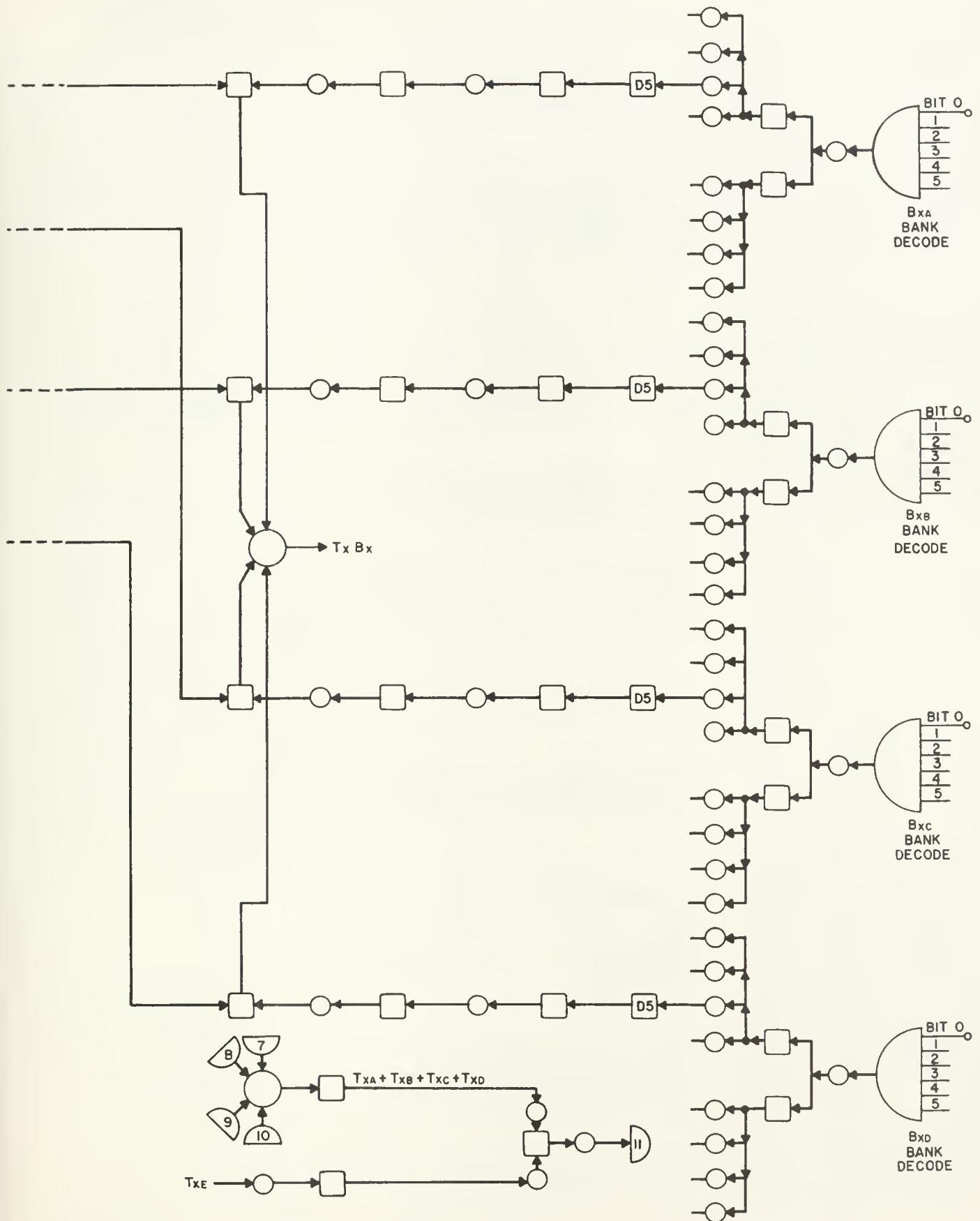


Figure 10. Trunk-Bank Selection (Sheet 2)

Since each CPU may enter data on the belt or accept data from the belt, by proper selection of instructions, one CPU may be made to communicate with another. Likewise, since CM may enter data on the belt or accept data from the belt, relocation of information with CM may be accomplished.

Data and address belts operate somewhat asynchronously. (See Figure 13). Data enters the belt only when the belt is free of pertinent information. The address belt detects whether a "store" or "fetch" tag is present and if neither is present generates a signal to clear both data and address belts. Additionally, the "address" belt detects at each stage whether the address on the belt is the same as that which was entered at the particular stage. If equivalence is found, then the data has made one trip around the belt and so can be removed from the belt. Therefore, it is possible to have data entering the belt at different stages and at different times and allow all CPU's to access this data at will.

Since the transmission time of the belt must be precisely maintained in order to control the flow of information on to and from the belt, each stage propagation delay must be maintained within rather narrow limits. This would be accomplished thru the use of a cable between stages whose length would be determined so as to maintain the necessary delay times. Additionally, the belt would be reclocked periodically, say every eight stages, so as to maintain proper pulse width, height, and timing. This pulse forming network would have a propagation delay equal to that of the delay line and therefore may be substituted for it in the chain.

Each stage of the data distribution belt is identical except for the signals applied to its terminal points. It is to be noted that the data flow path utilizes the same logic circuitry as that in the memory selection belt. Thus, both belts may be made sufficiently modular so as to facilitate minimum "down" time because of module failure. (See Figure 14.)

Each of the gates of the distribution belts are considered to have a propagation delay of 5 nanoseconds except for the "exclusively or" networks which have 10 nanoseconds. It is expected that such delays can be realized at the present time (see Fairchild CT μ L) and that shorter delay times will be realized in the future. Consequently, if we consider that gate propagation delays may uniformly be held to, say, 2 nanoseconds, then one trip around the belt can be accomplished in 256 nanoseconds. At this point, we have just about reached the limit of maximum instruction execution rates.

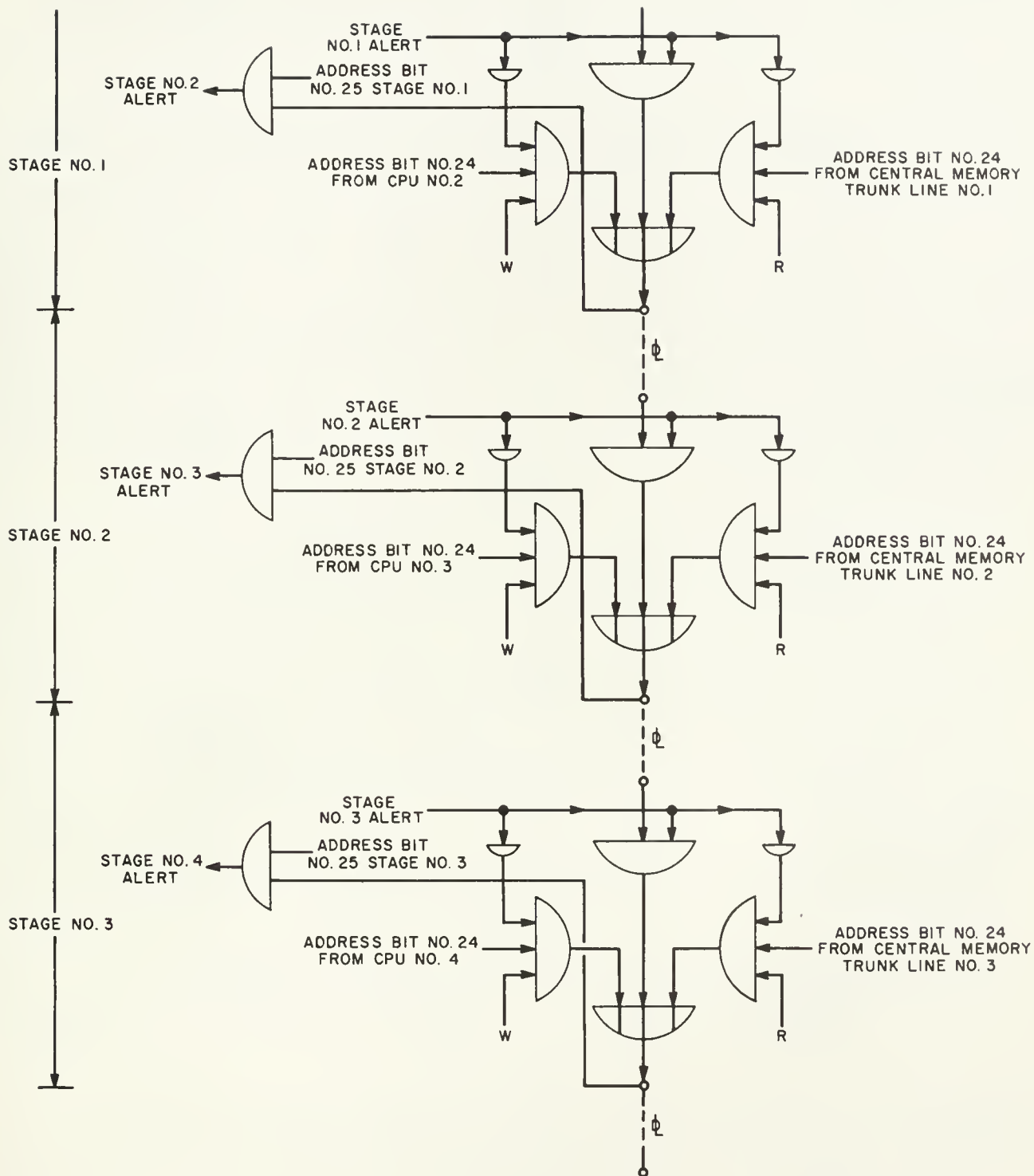


Figure 13. Data and Address Belt Alert Generation

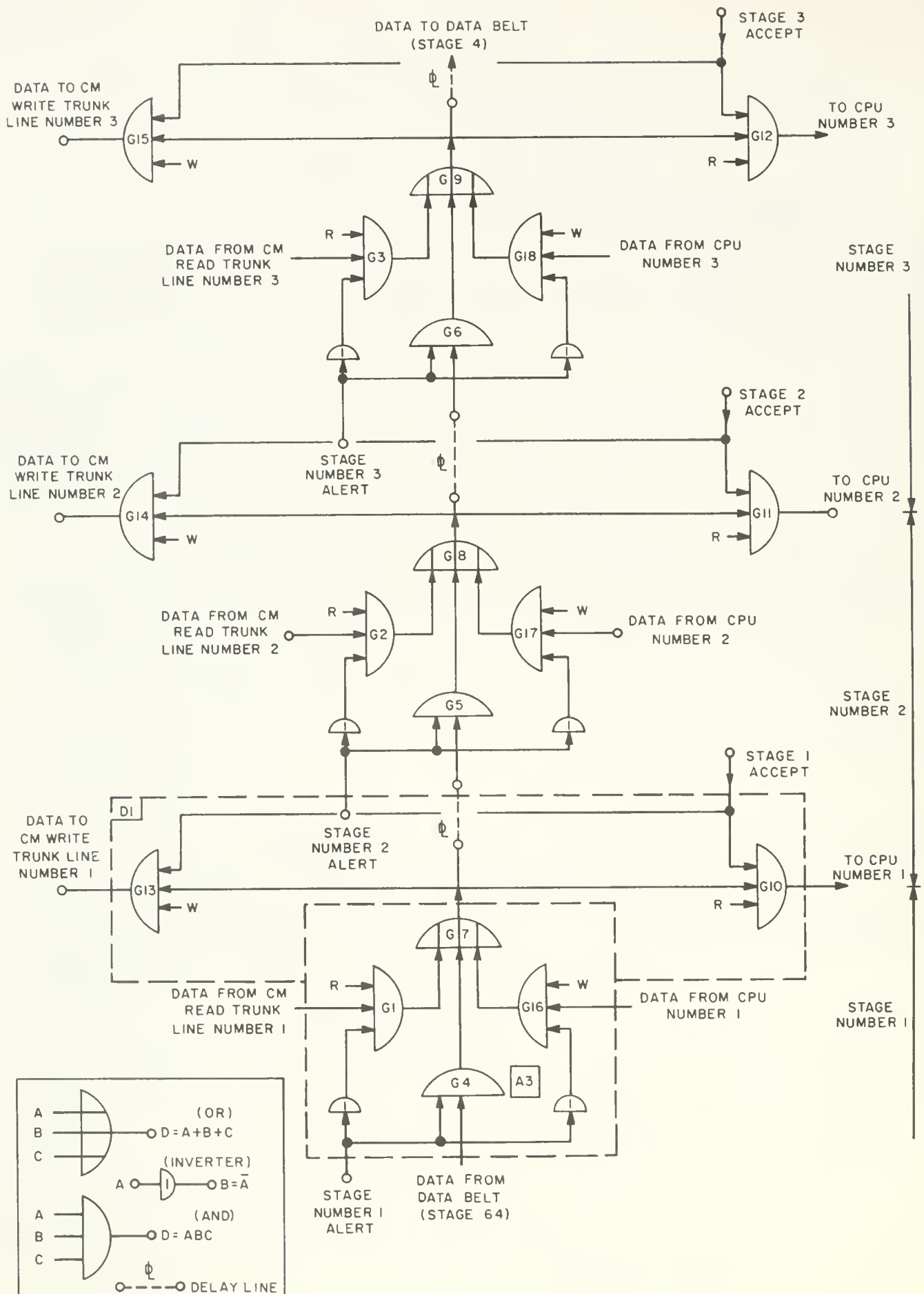


Figure 14. Data Distribution Belt (1 -Bit)

11. DATA AND ADDRESS DISTRIBUTION BELT OPERATION

With reference to Figures 11-14, assume that CPU No. 3 has called for information from CM reservoir No. 1 and that this data has been extracted and now appears on "CM read trunk line No. 1". At some time $t = -5$ a "stage 1 alert" signal had been generated so that G1 of the data belt allows the information to pass at $t = 0$ in the presence of a fetch signal (R). At the same time ($t = 0$), the address corresponding to this information is passed through G25 of the address belt. At $t = 10$, the data appears at the input to the delay line and is available to G10. The address is available to both "exclusively or" gates of stage 1 on the address belt. One of the exclusively or gates checks equivalence of the address on the belt with that requested by CPU No. 4 and since CPU No. 4 did not call for this data, no "stage 4" alert signal can be generated. The other "exclusively or" network checks equivalence with the address requested by CPU No. 3. Since this data had been called for by CPU No. 3, an "equivalence" signal out of the e.o. will result at $t = 20$. By this time, the data has passed through the delay line on the data belt and through G5. At $t = 25$ the data is available to G11 and the address is available to both e.o.'s of stage 2. The "equivalence" signal now is at the output of G31. Neither e.o. of stage 2 can generate an "equivalence" signal since neither CPU No. 4 nor CPU No. 5 have requested the data. At $t = 30$, the data has passed through the delay line of stage 3, the address has passed through its corresponding delay line, and the "equivalence" signal has reached the input of G34 having passed through an inverter which adds 5 nanoseconds of delay. At $t = 35$, the data is at the input to G9, the address is at the input to G24, and the "equivalence" signal is at the output of G34. At $t = 40$, the data is available to G12, the address is available to the e.o.'s of stage 3 and the "equivalence" signal is at the output of the inverter following G34. This signal is the "stage No. 3 accept" signal which appears at G12 of the data belt. Therefore, the data is passed through G12 and passes on to CPU No. 3 at $t = 45$, and the transfer of data from CM to CPU No. 3 has been accomplished. Neither e.o. of stage No. 3 of the address belt can generate an "equivalence" signal since neither CPU No. 5 nor CPU No. 6 have requested this data.

Both data and address continue around the belt in the foregoing manner until the address appears at the e.o.'s of stage 62. Here, address equivalence with that of stage No. 1 will be met and a "stage 1 alert" signal will be generated 25 nanoseconds later. During this time, the address and data have moved through two stages of the belt and now appear at G19 and G4 respectively. The "stage alert" signal is an inhibiting signal for these gates and so both address and data are prohibited from further travel around the belt. Hence, address and data have made one trip around the belt and have then been

removed. Five nanoseconds later G1 is alerted for new data from CM reservoir No. 1.

It is to be noted that the address "equivalence" check is made at every stage of the belt so that should two CPU's request the same data, it can be accepted by each. For instance, should CPU No. 4 request the same data as CPU No. 3, 15 nanoseconds after generating a "stage 3 accept" signal out of the inverter following G34, a "stage 4 accept" signal is generated out of the inverter following G35. Correspondingly, should all of the CPU's request the same data, each would receive it at 15 nanosecond intervals.

Assuming that each CPU can make individual requests for memory, it is possible for each CPU to address one of 64 CM reservoirs. Consequently, the data belt and address belt would be carrying 64 words of data and 64 addresses simultaneously each word and address appearing at each CPU at 15 nanosecond intervals. If the CPU were capable of accepting information at this rate, an effective data transfer rate of 4.26×10^9 words/sec would be accomplished.

In order to accomplish a CPU-to-CPU transfer, it would be necessary for a CPU to call for data from another CPU through its instruction code. As a result, instead of CM data being entered on the belt, data from the requested CPU would be entered along with its identifying address. The procedure for retrieval of the data by the requesting CPU would then follow that of retrieval of data from CM. Thus, one CPU may communicate with another.

A memory-to-memory transfer may be accomplished by one CPU requesting data from CM (fetch) and then writing it into CM (store). The minimum time for such a transfer could be 90 nanoseconds.

12. DATA DISTRIBUTION BELT LIMITATIONS

Operating the conveyer belt at its maximum efficiency implies that 64 words may be placed on the belt every microsecond. This is true because of the restriction that data placed on the belt must make one complete trip before being removed. Hence, if the belt is full, no further data may be inserted until there is a free "slot". This cannot occur until data is removed from the belt at the end of its trip. On this basis, a CPU request for memory data (fetch) may be made only once every microsecond. However, a CPU may remove data from the belt every 15 nanoseconds. Therefore, 64 words may be transferred to any CPU from the belt within one microsecond. As far as the

CPU is concerned, this is an effective transfer rate of 64×10^6 words/sec. (or 4.096×10^9 bits/sec.)

If we assume that only one CPU requests data from central memory then, providing the requests are made from a single reservoir (but from different banks within the reservoir), data can enter the belt at 15 nanosecond intervals. Again, 64 words can be fetched from memory within one microsecond. If the processor attempts to address successive reservoirs, then a delay in entering the belt will be experienced because of previous data already on the belt. For instance, assume CPU No. 1 requests central memory data from reservoir No. 1. This data is entered on the belt at some time $t = 0$. Fifteen nanoseconds later CPU No. 1 requests central memory data from reservoir No. 2. This data would be entered on the belt at $t = 15$ from trunk line No. 2 except that the previously requested data has traversed one stage of the belt in the 15 nanosecond interval and now appears at stage 2. Since the belt always has priority, the second request cannot enter until the first request has traversed stage 2. This takes another 15 nanoseconds and therefore request 2 enters the belt at $t = 30$. At $t = 30$ another request is made and would normally be entered at stage 3. However, data No. 1 is at stage 3 and data No. 2 is in stage 2. Therefore, data 3 must wait 30 nanoseconds before it can be entered on the belt. This occurs at $t = 60$. At this time data No. 1 is at stage 5, data No. 2 at stage 4, and data No. 3 at stage 3. Data No. 4 is requested at $t = 45$ but data No. 1 is at stage 4 backed up by data No. 2 and data No. 3. Therefore, data No. 4 cannot get on the belt until $t = 90$. This continues until request No. 32 is made which cannot enter the belt until $t = 960$. Request No. 33 enters at $t = 995$. As a result, the effective data transfer rate has been halved. (33×10^6 words/sec.)

It turns out that no matter how the memory is addressed by a single CPU the best data transfer rate would be 64 words every microsecond and this only if the CPU addresses successive banks of a single reservoir.

Two CPU's addressing two data trunks simultaneously at 15 nanosecond intervals again can obtain an effective data rate of only 64 words per microsecond. With two CPU's working simultaneously, the belt will be filled in 32 requests from each CPU. This would take only 480 nanoseconds (a transfer rate of 128 words per microsecond) but then the CPU's must wait another 480 nanoseconds before the 33rd request is entered on the belt. Hence, the effective transfer rate is still only 64 words per microsecond.

Similarly, no matter how many CPU's make requests simultaneously, an effective transfer rate of no better than 64 words per microsecond may be obtained for a single belt system.

In order to increase the rate of data flow from CM to CPU either data must be transmitted along the belt at faster rates than 15 nanoseconds per stage, or each CPU must be capable of looking for data at more than one place on the belt.

As has been indicated, propagation time per gate will probably not be reduced below 2 nanoseconds. Therefore, the best data transfer rate to be expected will be 64 words every 256 nanoseconds or 250×10^6 words per second. This is about one-fourth the desired rate of 10^9 words per second.

The use of multiple belts would increase the flow rate, but would add to the cost of the system. It is estimated that a single belt system for data distribution would cost approximately \$57,000. In order to increase the data flow rate to 10^9 words per second, 16 belt systems would be required at a cost in excess of \$900,000. Even if the cost of the memory system were to reach $\$10^6$ this would represent a nominal sum in light of the size, flexibility, and data flow rates that may be achieved. If core stack cost were to remain at \$0.05 per bit, the cost of the electronics is only 2% of the cost of the core memory.

13. CONCLUSIONS

Investigation of the problems attendant to the simultaneous selection of multiple words of information contained in a large capacity, central memory and simultaneous distribution of this information to appropriate processing units has indicated the feasibility of such a system from an engineering standpoint. Feasibility is predicated upon specifications outlined here for reference:

- 1) Central memory will contain 16,777,216 words of information.
- 2) Each word will be 64 bits in length.
- 3) Central memory will be organized into 4096 individual banks which will be arranged in a 64x64 array and each bank will contain 4096 words.
- 4) Processing units will number 64, and each will have access to any word in memory.
- 5) Processing units will have a "turn-around" time of 60 nanoseconds so that data transfer rates of 1.07×10^9 words/second may be achieved.

Primary effort has been directed toward the logical design of a memory addressing system which will conform to the outlined specifications. However, the design

is predicated on the assumption that memory addressing will not be under control of an executive program nor will any memory-bank/processor assignment be made. Therefore, it is not expected that a transfer rate of 1.07×10^9 words/sec. will be achieved.

In order to reduce the number of necessary components, a "time-sharing" scheme for processors is proposed. The 60 nanosecond turn-around time period is broken up into four equal "slots". Thus, 16 processors per time interval can address the memory. This allows each processor to have a 15 nanosecond memory select "slot" time and 45 nanosecond "compute" time. To achieve a 60 nsec. "turn-around" time, it is necessary that propagation delays of 5 nanoseconds per gate be obtained and maintained rather precisely. This may be achieved through the use of integrated circuitry now available from manufacturers (i.e. Fairchild CT μ L, and Motorola "MECL").

The "time-sharing" scheme for memory selection is illustrated in Figure 5. Thirty-two processors are permitted to enter their requests simultaneously onto a "belt" network. Sixteen of these requests are processed in the first "slot" interval while the remaining sixteen requests are moving along the belt. These requests are processed in the second interval. Thirty-two more requests are entered onto the belt at the third interval with sixteen of those requests being processed in the fourth interval. The sequence then repeats with the original thirty-two processors issuing new requests. In this manner, every processor is permitted to make memory requests at 60 nanosecond intervals.

The "belt" configuration lends itself to conform with physical distances which may occur between processing units. The required five nanosecond delay may be realized with a co-ax cable which is approximately 3.5 feet long. (Co-ax cable RG62/U has propagation delay of 1.2-1.5 nanoseconds/ft.)

The "belt" is continuous, but conflicting address requests remain on the belt only as long as the conflict exists. When conflicts exist, requests are transferred to another stage on the belt. Hence, a priority system is established for resolving conflicts. This priority system is in descending order of processing units.

Three types of conflicts can occur in the proposed system. The first type, which is called Class I, results from the distribution scheme proposed in the last Progress Report. That report described a single "trunk" line for every 64 banks of memory. Therefore, Class I conflicts occur when in a memory "slot" interval, two or more processors request information from different memory banks assigned to a single "trunk" line. In this instance, requests must be separated by 15 nanosecond intervals so as to conform to the "trunk" line information handling capabilities.

Class II conflicts result from the selection, within a memory "slot" interval, of a single bank of memory by two or more processors. Such requests must be separated by a memory cycle time since the condition is one in which an attempt is made to extract two or more words from the same bank of memory.

Class III conflicts can occur as the result of addressing the same bank of memory in an interval of time less than the memory cycle time. In this respect, Class II and Class III conflicts are similar. However, unlike Class II conflicts which necessitate specific request separation delays, request separation delays necessitated by Class III conflicts vary according to the number of memory "slot" intervals between requests. A scheme for resolving Class III conflicts is not included in this report.

Foremost in the process of resolving conflicts is their detection. Conflicts may be broken into two parts; those with trunk conflicts but no bank conflict (Class I), and those with both trunk and bank conflicts but no word conflict (Class II). (Class III conflicts are not under consideration at this time.)

It can readily be seen that a Class II conflict cannot exist without a Class I conflict. Consequently, it is reasonable to allow any resolving scheme to be dependent upon the detection of a Class I conflict or more specifically upon the detection of a trunk conflict.

It can readily be shown that if no trunk conflict exists, then different banks of memory are being addressed and therefore, each request may be processed. However, if a trunk conflict exists, then the class of conflict must be determined, and a priority established so as to process one request in preference to the other. Additionally, the unprocessed request must be reissued at some future time.

Establishment of priority in the proposed system is arbitrary and has been chosen so that the "belt" appears to be continuous. In this manner, the unselected requests "shift" into succeeding sections of the belt.

The priority signals generated are combined with the appropriate bank decode outputs to obtain a trunk/bank selection. Thus, $TxBx = TxA Bxa + \bar{T}xA Txb Bxb + \bar{T}xA \bar{T}xb Txc Bxc + \bar{T}xA \bar{T}xb \bar{T}xc Txd Bxd$. It can be seen that through this artifice the proper memory bank has been selected. It is also evident that 4096 such trunk/bank combinations must be established.

As yet, no scheme has been devised for the selection of the proper word within the selected bank nor for the detection of Class III conflicts. It is felt that any scheme

for the selection of the proper word will be dependent upon the memory organization (i.e. linear select, bit oriented, etc.). Further investigation into the detection and resolving of Class III conflicts will be made.

A cost estimate for the proposed system has been made based upon the use of integrated circuitry and present-day prices. The use of integrated circuitry would enhance the uniformity of circuit parameters so as to maintain consistent propagation delays which are essential to the system. It would also help to conserve space thereby reducing transmission times. It is not expected that cost reduction in the manufacture of integrated circuitry will take as sharp a decline as it has in the past. However, it is expected that manufacturers will place emphasis on larger logic arrays and increased speeds. This expectation is bolstered by conversations with representatives of various manufacturers. Therefore, if the necessary components become available, it is estimated that the memory selection and distribution system, would cost approximately \$10⁶. In the light of such a large project as proposed, this sum is nominal.

It is evident from the investigation that the required specifications may be achieved only if the 64 processing units request 64 different banks within a memory cycle time. (At this time, only memory cycle times of 1 microsecond are under consideration.) Should two or more processing units request information contained in a common memory bank within this time, then transmission conflicts occur which can significantly reduce the data transmission rate. Additionally, processing units should be permitted to make requests for memory information independently or whether or not previous requests have been acknowledged. (If a philosophy of "request-receive-request" is adhered to, then the "turn-around" time for a processor exceeds 60 nanoseconds unless memory cycle times are in the order of 20-25 nanoseconds. This is evident when co-ax transmission delays, memory cycle times and circuit delays are accounted for.) Therefore, software should consider the practicality of memory requests by one processor for use by another processor. In this manner, each processor may achieve its required "turn-around" time and maximum transfer rates may be achieved.

It is also evident that unless software resolves the problem of conflicts, the probability of achieving maximum data transfer rate of 10⁹ words per second is negligible. (If all of the processing units are permitted to address memory simultaneously with the probability of any unit selecting a particular address being $1/\eta$, then the probability of two units selecting the same address will be $1/\eta^2 \times aC_2 = 1/4096 \times \frac{64!}{2!62!}$
 $= \frac{64 \times 63}{2 \times 4096} = \frac{4032}{8192} = 0.49$. This implies a conflict every 120 nanoseconds which, assuming that conflicts may be resolved within 60 nanoseconds, will reduce information transfer

rates to 0.54×10^9 words/sec.) It is obvious that if account is taken of the probability of two or more unit requests for the same memory bank, that conflicts are an almost certainty. It is therefore to be concluded that data transfer rates of 1.07×10^9 words/sec. cannot be achieved unless 1) a philosophy of "request-receive-request" is discarded and 2) processing units make requests under control of an executive program so as to eliminate conflicts or are assigned specific banks with which to communicate.

It is not expected that the cost of integrated circuits will experience as sharp a decline in the next decade as it has in the past two years. However, some reduction in cost can be expected as manufacturers gain larger yields through improved technology. On the other hand, since we are demanding greater speeds for our system, it is conceivable that these increases in yield would be offset. As a result, no large reduction in cost would obtain unless significant breakthroughs are made in device technology and manufacturing processes.

It is appropriate to make some estimates on the cost of the memory itself. At present-day prices, 4096 words of core memory costs in the neighborhood of \$15K to \$20K. This fixes the cost per bit at about \$.05 to \$.06. These prices can be significantly reduced if very large quantities are purchased. Assuming the cost per bit to be reduced to \$.01 because of the large quantity required for the proposed system, then the cost of a 16 million word memory would run in the neighborhood of 10 million dollars. This does not include the electronics to run this memory. If this expenditure represents say one-third to one-half the total cost of the memory system (which is not unlikely), then we are talking about an expenditure of 20 to 30 million dollars for a memory system. This does not include costs for such things as CPU's, display systems, controllers, etc.

BIBLIOGRAPHY

- (1) J. T. Schwartz, "A Proposal for an Investigation into the Design and Use of Very Large Parallel Computers", Unpublished Proposal (1965), Courant Institute of Mathematical Sciences, New York University.
- (2) Department of Computer Science, University of Illinois, Urbana, Illinois, "Proposal for Experimentation in Parallel Computation", October 7, 1965, (submitted to Advanced Research Projects Agency).
- (3) J. T. Schwartz, "Large Parallel Computers", Unpublished Paper, Courant Institute of Mathematical Sciences, New York University (1965).
- (4) Private Conversation with Professor J. T. Schwartz, Courant Institute of Mathematical Sciences, New York University (1965).
- (5) J. A. Rajchman, "Magnetic Memories-Capabilities and Limitations", Journal of Applied Physics, Vol. 34, No. 4 (Pt. 2), April 1963.
- (6) Progress Report No. 33, Unpublished Report, Engineering Research Laboratory, Courant Institute of Mathematical Sciences, New York University, February 26, 1965.
- (7) Rice, "Computers of the Future", Proceedings of the Eastern Joint Computer Conference, 1959.
- (8) Louis and Shevel, "Storage Systems-Present Status and Anticipated Development", IEEE Transactions on Magnetics, Vol. Mag-1, No. 8, September 1965.
- (9) Westinghouse Electric Corporation, "Survey of Highly Parallel Information Processing Systems", prepared for Advanced Planning Division, Naval Analysis Office of Naval Research, 1964.
- (10) R. Laupheimer, "An Improved Union of Digit-Select Lines and Sense Lines in Word Select Ferrite-Core Memory Systems", Proceedings of the IEEE, Vol. 52, No. 4 (April 1964).
- (11) AEC Computing and Applied Mathematics Center, Courant Institute of Mathematical Sciences, New York University, "Activities of the Engineering Research Laboratory July 1, 1963-June 30, 1964", AEC Research and Development Report, NYO 1480-2, August 1, 1964, pp. 36-37.

This report was prepared as an account of Government sponsored work. Neither the United States, nor the Commission, nor any person acting on behalf of the Commission:

- A. Makes any warranty or representation, express or implied, with respect to the accuracy, completeness, or usefulness of the information contained in this report, or that the use of any information, apparatus, method, or process disclosed in this report may not infringe privately owned rights; or
- B. Assumes any liabilities with respect to the use of, or for damages resulting from the use of any information, apparatus, method, or process disclosed in this report.

As used in the above, "person acting on behalf of the Commission" includes any employee or contractor of the Commission, or employee of such contractor, to the extent that such employee or contractor of the Commission, or employee of such contractor prepares, disseminates, or provides access to, any information pursuant to his employment or contract with the Commission, or his employment with such contractor.

DATE DUE

PRINTED IN U S A.

NYU-NYO-
1480-48

c.1

Ruben

NYU-NYO-
1480-48

c.1

of
b.

NYU-NYO-
1480-48

c.1

AUTHOR

Ruben

TITLE

Engineering aspects
of high speed data distrib.

DATE
DUE

BORROWER'S NAME

ROOM
NUMBER

**N.Y.U. Courant Institute of
Mathematical Sciences**

251 Mercer St.
New York 12, N. Y.

